Enabling Domain Experts to Model and Execute Tasks in Flexible Human-Robot Teams

Dominik Riedelbauch, Tobias Werner, and Dominik Henrich

Chair for Robotics and Embedded Systems,
Universität Bayreuth, D-95440 Bayreuth, Germany,
dominik.riedelbauch@uni-bayreuth.de,
http://robotics.uni-bayreuth.de

Abstract. Recent advances in safe human-robot coexistence make collaboration of humans and robots in achieving common goals feasible. We propose a concept that treats human and robot agents as equal partners in executing a task specified by a shared task model. Equality between agents offers high flexibility, as e.g. the team composition may change arbitrarily without interrupting the working progress. The main challenge in achieving flexible teaming is coordinating the robot with operations executed by human partners. We contribute an approach to this problem that is based on observing pre- and postconditions of operations using a robot-mounted camera system. The coordination mechanism is embedded into a framework that allows domain experts to create, test, and dispatch new task models to collaborative execution. The approach is validated by experimental tasks composed of pick-and-place operations.

Keywords: human-robot collaboration, equal-partners teamwork, human-robot coordination, task modelling

1 Introduction

Traditional robot systems are designed to repeatedly perform the same task for a long period of time. The safety of human workers in industrial setups is often ensured by fences that prevent them from entering the robot workspace. Recent advances in the field of sensing enable safe human-robot coexistence without these physical barriers (e.g. by fast reconstruction of the robot workspace [26] and reactive motion planning [5] or tactile sensors [2]). We envision a hybrid assembly cell that follows the paradigm of symbiotic assembly [3] and thus uses the precision and strength of robots as well as the cognitive skills unique to humans to work together on achieving a common goal. By symbiotic combination of intelligent robotics and human knowledge and experience, systems that offer the flexibility required for applying robots in small batch production, handicraft workshops or small laboratories can be designed. We therefore propose a method that combines execution of robot operations contributing to goal achievement with operations to acquire sensor data needed for synchronizing with human actions. This mechanism is embedded into a system allowing domain experts to model and execute tasks in human-robot collaboration.

2 Related Work

We structure work related to human-robot common goal achievement based on task models according to the degree of flexibility they offer to the participating agents during task execution. Least flexibility is offered by systems that perform a **fixed assignment of operations** within a task to agents in advance of the execution process. In [10], methods from classical assembly planning are adapted to build a hierarchical framework for optimal task allocation and execution in human-robot collaborative assembly. The HRI/OS [4] uses a centralized executive to delegate tasks to humans and robots. Groups of operations can also be allocated to agents by assigning a role to each of them [13][21].

The next level in our taxonomy is formed by approaches where either a **human or robot dominates** the process, while the counterpart within the team follows orders or adapts. Control can be given to the robot, which plans, assigns and explains parts of the task to a human partner [1] [20]. The task allocation is negotiated in [7], where a robot asks the human for permission before performing an operation. Other approaches regard the robot as a tool that assists humans by performing assistive actions [6][12][16].

We regard systems with a focus on **decision authority for all agents** as the most flexible. The Chaski executive [24] enables dynamic execution of tasks formulated as Temporal Constraint Networks (TCNs) by just-in-time assignment of operations to agents. Each agent decides for the next operation on the fly based on the decisions taken and communicated by others, resulting in an execution process that fulfils all time constraints encoded in the TCN. Similar to [24], the approach of [17] uses precompiled versions of Temporal Planning Networks under Uncertainty for integrated plan recognition and dynamic execution.

Our hypothesis is that a robot system should possess the properties listed below to achieve symbiotic collaboration. These properties also allow the integration of our approach into the above taxonomy. An exchangeable *task model* should be the main input, preferably one similar to those used in assembly planning (e.g. precedence graphs, AND/OR-Graphs [8]) so that existing planning algorithms can be reused for adding knowledge to the system. The task model is *shared* between humans and robots to approximate a *shared mental model* [11]. Studies on human teaming show, that this increases team work effectiveness [18][19]. It may be necessary for human agents to handle interrupts in small workshops, e.g to serve entering clients. If the system allows for a *dynamic team setup*, meaning that the number of humans and robots can arbitrarily change, the working progress will not be stopped in such cases. This can not be achieved by a pre-computed schedule, as task allocation needs to vary depending on the current team composition. Thus, requesting flexible teams requires *dynamic plan execution*, where agents decide about their next steps on the fly [24]. Combining flexible teams and agents with strongly asymmetrical capabilities would likely stop the progress when highly skilled agents leave. We therefore focus on tasks composed of operations that can be executed by either human or robot, possibly within differing amounts of time, and refer to this as *equal partners collaboration*. Our understanding of equal partnership is based on [24], where *equal*

partners teamwork means, "that each member of the team has equal authority to make decisions when executing the plan". Through this paradigm, intuition of humans about their own capabilities and preferences is utilized to achieve a workflow that is convenient for workers, e.g. regarding ergonomics. We explicitly focus small-scale scenarios, where flexibility is favored over time optimal task allocation.

As our coordination mechanism is designed to satisfy the above criteria, we sort it into the category of approaches with decision authority for all agents in our related work taxonomy. While [24] and [17] focus on fulfilling timing constraints, we explicitly enable flexible team composition. In [24], the decisions are taken based on other agents' communication, and [17] receives estimates of the world state from an external component. In contrast, our contribution plans sensor operations in addition to task operations to gather information needed for robots to select their next actions just-in-time.

3 Coordinating Flexible Human-Robot Teams

Our approach is based on *elementary operation templates* $O = \{o_1,\ o_2,\ ...,\ o_{|O|}\}$ that the robot is able to execute. Every operation template $o \in O$ needs parameters taken from the set P_o of valid parameter combinations to o. E.g., a pick-and-place operation may need start and goal position of an affected object. A pair $\bar{o} = (o,\ p)$, $p \in P_o$ is named an *operation instance*. O partitions into two subsets O_S and O_T. O_S is a set of *sensor operation templates* that are used to acquire sensor data, e.g. moving a camera and capturing an image. The *task operation templates* O_T are used to form shared task models. A *shared task model* $T = \{\bar{o}_1,\ \bar{o}_2,\ ...,\ \bar{o}_{|T|}\}$ is composed of several operation instances whose templates are taken from O_T. The elements of T may be part of some superordinate structure like the graph defined by precedence relations [8]. An exemplary model of a palletizing task as used in our experiments is depicted in Fig. 1.

The coordination mechanism bases upon the observation, that elementary operations usually need some *preconditions* to be satisfied before they can be executed. E.g., a pick-and-place operation requires the affected object to exist within the workspace. Preconditions we are considering are not limited to resources, as the presence of humans in the workspace might result in additional requirements. E.g., if a task includes a welding operation, robot agents need to ensure that the safety door of the welding system is closed. As soon as an operation was successful, it produces observable effects manifesting in fulfilled *postconditions*. E.g., the moved object will be at it's new position. More formally, we define the sets V_o of *pre-* and N_o of *postcondition templates* for every template $o \in O_T$. A *condition instance* $\bar{c} = (c,\ \bar{o})$ combines a condition template with the operation instance $\bar{o}$ it has to be evaluated for. Every operation $\bar{o}_i = (o_i,\ p_i)$ within a task T has a set of *preconditions* $V(\bar{o}_i)$ that contains an instance for every template in V_{o_i}. The postconditions $N(\bar{o}_i)$ are defined respectively.

A set of fulfilled pre- or postconditions allows to decide whether the corresponding operation may be started or has already been completed. Hence,

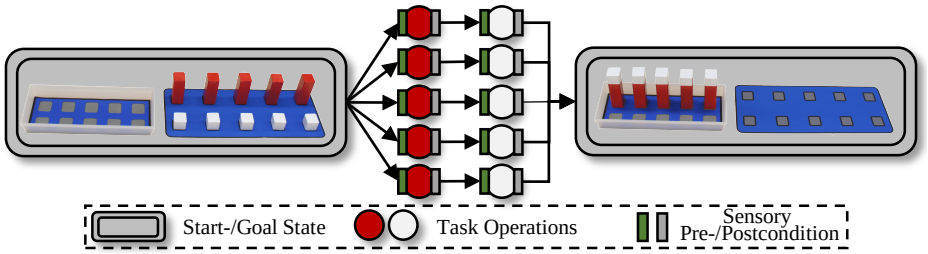

Fig. 1. Tasks can be modelled as precedence graphs that convert a start state into the desired goal state by a suitable composition of task operations. Each operation is annotated with pre- and postconditions needed for coordinating humans and robots.

perceiving the state of conditions provides a way to decide whether an operation is executable. We model the process of condition evaluation as a function $\mathcal{P}$ that abstracts the details of perception. $\mathcal{P}$ takes an arbitrary set of condition instances and returns **true**, if all of them are fulfilled, otherwise **false**. Internally, $\mathcal{P}$ may trigger sensor operations to gather suitable data for the evaluation. Given $\mathcal{P}$, *readiness* $\mathcal{R}$ and *success* $\mathcal{S}$ of an operation instance $\bar{o}_i$ are defined by

$$\mathcal{R}(\bar{o}_i) = \mathcal{P}(V(\bar{o}_i)) \qquad \mathcal{S}(\bar{o}_i) = \mathcal{P}(N(\bar{o}_i)).$$

If and only if $\mathcal{S}(\bar{o}_i) = \textbf{true}$, then $\bar{o}_i$ has already been done. Respectively, if and only if $\mathcal{R}(\bar{o}_i) = \textbf{true}$, all preconditions of $\bar{o}_i$ are fulfilled. An operation instance with $\mathcal{R}(\bar{o}) = \textbf{true}$ and $\mathcal{S}(\bar{o}) = \textbf{false}$ is named *active*. Correct task execution is guaranteed by only executing active operations.

4 Prototype System

In the following, we describe components of our prototype system based on the above coordination mechanism in detail. The main modules are depicted in Fig. 2. The workflow consists of three steps: First, the task model is created using a graphical editor (Fig. 3, left). The editor offers functionality to add task objects (blue boxes) and operations (gray boxes) working on them. Required operation parameters are input through dialog windows. A task structure is created by connecting operations using precedence graph edges. The results of task modelling can be checked by observing the robot performing the task within a simulation (Fig. 3, right). Erroneous operations or parameters can be corrected by returning to the editor. The precedence graph undergoes an automatic preprocessing step that supplements each operation with suitable pre- and postcondition instances. Then, the graph is stored as an XML file for later usage and can be passed to a module for collaborative execution. Given a specific set of sensors and actuators, this module encapsulates an algorithm to determine an efficient order of condition validation steps and execution of operations, such that the task is carried out correctly. Details of our current implementation of the execution module are described in Sections 4.1, 4.3 and 4.2.

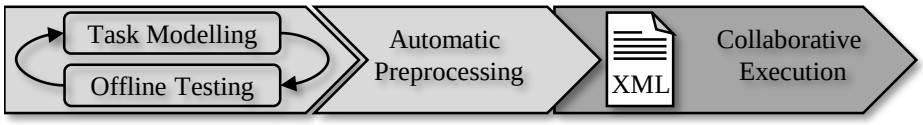

Fig. 2. The system workflow has three steps: Task models are created by repeated modelling and offline testing. A preprocessing step generates required conditions. This results in an XML task representation serving as input to the execution module.

4.1 Hardware, Supported Operations and Conditions

We use a KUKA LBR IV with a Robotiq 3-Finger gripper. Worker safety is ensured by using the robot in compliant mode and at moderate speed. Though our approach allows integrating arbitrary sensors, we focus on cameras for condition evaluation. In theory, the approach described in Section 3 could be implemented by equipping the robot cell with cameras overseeing the whole workspace at any time, and repeatedly evaluating all conditions e.g. through object recognition and localization. The set of active operations would then always be available for the system to select it's next step. We intend to manipulate small objects within a workspace, where several humans and robots might be moving. Thus, a high level of occlusion may be expected when using fixed cameras, rendering them impractical. Therefore, the robot is equipped with an IDS uEye UI-1220SE-C-HQ eye-in-hand camera, enabling it to look at specific positions for condition evaluation. Besides the problem of occlusions, this also reduces the amount of hardware within the workspace, making the installation less complex and costly.

Our set of operation templates O_T enables manipulation tasks through operations to transfer objects from fixed start to goal positions and apply a stamp to them. In this context, condition templates to evaluate, whether an object of a certain type is present at a given position or not, and whether it is stamped or not, are needed. The only required sensor operation o_S moves the robot to a given position and triggers the camera. The perception function $\mathcal{P}$ maps each input condition instance to an instance of o_S. Condition evaluation is realized through object recognition applied to the resulting image, specifically by classifying objects according to their color. Our approach can intuitively be extended to more complex scenarios by adding hardware (e.g. a tool changer), operations (e.g sensor-guided skills [25]), sensors and evaluation algorithms (e.g. active manipulation for recognition of objects [22] and material properties [28]). However, such special implementations are out of the scope of this conceptual paper.

4.2 Execution Module Software Design

The execution module is structured as a multi-agent system [9]. A *coordination agent* executes the algorithm to achieve correct task execution through task and sensor operations as described in Section 4.3. To this end, it requests evaluation of $\mathcal{R}$ and $\mathcal{S}$ from the *perception agent*, which maps the conditions to sensor operations and performs image processing. Task operations as well as sensor

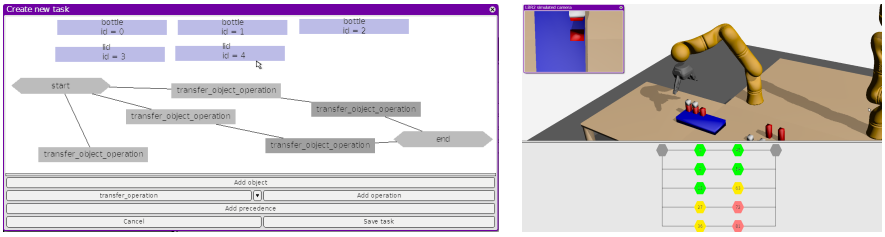

Fig. 3. The prototype system provides a graphical editor for modelling new tasks (left). The execution of tasks by a robot can be observed in a simulation environment (right) to validate correct parametrization of elementary operations.

operations are dispatched to the *hardware agent* that transforms abstract operations into concrete hardware commands.

The execution module realizes all agents through the ENACT software framework [27]. This framework follows the popular programming paradigm of minimized coupling and maximized cohesion (see [14]). To this end, ENACT offers exchangeable agents and couples these through abstract data types. This enables quick porting of the execution module to different hardware. For instance, changing the robot type is intuitively done by implementing and using another hardware agent. The same holds for new sensors and the perception agent.

4.3 Execution Algorithm for Robot-Mounted Cameras

Cameras attached to the robot can only view parts of the workspace at a time. The challenge lies in finding an order of condition evaluation and execution of operations that reduces the number of sensor operations, while still ensuring correct task execution. This can be achieved by tracking the task progress to extract operations and conditions currently relevant. We therefore integrate operations into precedence graphs (Fig. 1). The knowledge a robot has about the progress is encoded in two sets I_1 and I_2. I_1 denotes operations whose predecessors in the graph have successfully been executed. Thus, elements of I_1 are candidates for testing whether they are active. I_1 is initialized with all operations directly connected to the start node. Operations with preconditions that have been evaluated negatively and consequently are assumed to be carried out by another agent at the moment are held in I_2. Initially, I_2 is empty. The execution algorithm is visualized in Fig. 4. As long as I_1 or I_2 contain any elements, the task is not completed and the algorithm keeps running. In the *Execution Phase* (dark grey), an element $\bar{o}$ from I_1 is selected first. Currently, the selection is biased towards following branches within the precedence graph, meaning that completing subtasks is preferred. Further concepts like a capability index [23] may be integrated to e.g. prefer operations with a high automation potential. If the postconditions of $\bar{o}$ are fulfilled ($\mathcal{S}(\bar{o}) = \mathbf{true}$), all successors of $\bar{o}$ whose other predecessors already have been completed can be added to I_1. Otherwise, preconditions are checked by determining $\mathcal{R}(\bar{o})$. If they are fulfilled, $\bar{o}$ is dispatched to the execution agent.

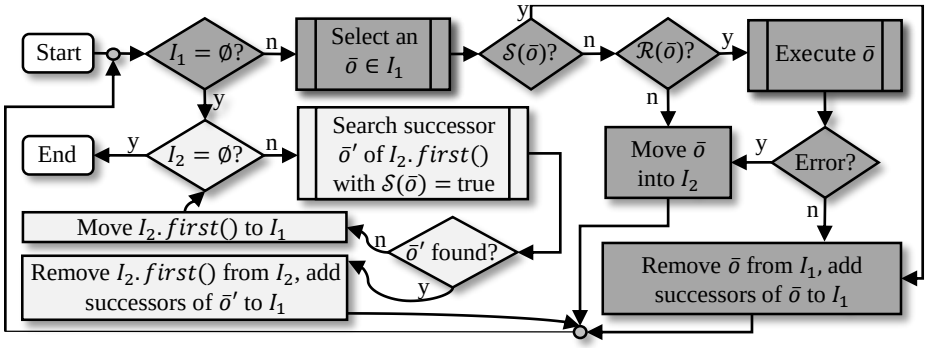

Fig. 4. Our algorithm for human-robot collaboration has two phases. In the *Execution Phase* (dark grey), the system looks for active operations and executes them. If operations with unfulfilled preconditions are detect, they are stored for further processing in the *Monitoring Phase* (light grey) that tries to detect progress and may move operations back to the Execution Phase for a retry.

We currently assume, that no errors occur during execution. This assumption can be relaxed using robust operation implementations that support rollback on errors [15]. If preconditions are not fulfilled, $\bar{o}$ is assumed to be in progress and moved to I_2 for later clarification. The Monitoring Phase is entered as soon as there are no tasks left in I_1 - the system has to synchronize with operations executed by humans during the Execution Phase. To this end, the elements of I_2 are iterated by repeatedly processing the first element $I_2.first()$ of I_2. If the postconditions of an $\bar{o} \in I_2$ are fulfilled, the successors of $\bar{o}$ are up next and can be moved into I_1 for the next Execution Phase. In addition, postconditions of the successors of $\bar{o}$ are checked. This enables the system to "catch up" and detect completed operations without having spotted their unfulfilled preconditions in the Execution Phase. This initial realization is based on the assumption, that humans participating in the collaboration are cooperative and will carry out the task correctly due to the task model. For pick-and-place operations, this means that an object that is not detected at it's start position will reappear there or will be moved to it's target eventually.

5 Experimental Validation

Exemplary tasks as used in our experimental validation are shown in Fig. 5. The experiments are targeting the evaluation of the equal-partners collaboration mode using our coordination mechanism. Therefore, we are using simply shaped, colored objects to abstract from the problems of grasp planning and object recognition. We used the GUI (Fig. 3) to create and test task models simulating assemblies (1), palletizing tasks (2) and processes involving tools like a stamp by combining the pick-and-place operations described in Section 4.1. Task 2 emulates palletizing bottles (red blocks) and sealing them with lids (white blocks). The complete model of Task 2 is shown in Fig. 1. Some steps during

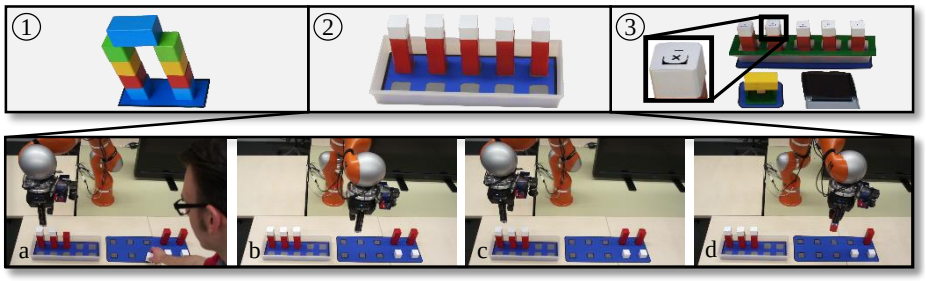

Fig. 5. Our experiments consist of setups simulating assembly (1), scalable palletizing tasks (2) and processes involving tools in addition to operating resources (3). Crucial steps during an execution of Task 2, showing how the robot detects human interaction and proceeds working, are depicted in the lower part of the figure.

an execution process are shown in the lower row of Fig. 5: The human picks up a lid, while the robot negatively evaluates the postcondition for the pick-and-place operation afflicting the same object (a). The robot moves on to check the precondition of this operation (b). As it is not fulfilled, the execution algorithm continues with another operation contained in the list of elements I_1 that are up next according to their precedence relations. In the sample case, the robot looks for the next bottle at it's target position (c). As it is not present, the precondition is checked moving to the start position and trying to locate it there. This evaluation step is successful, allowing the robot to perform the operation (d).

6 Conclusion and Future Work

We presented an approach to equal-partners human-robot collaboration that enables coordinating teams of flexible composition. The approach is based on evaluating pre- and postconditions of operations to detect human actions. It combines dynamic execution of robot operations contributing to goal achievement with operations to acquire sensor data. We integrated it into a framework enabling domain experts to create and test shared task models using a Graphical User Interface. This system can be adapted to tasks composed of arbitrary elementary operations by adding suitable sensors, actors and condition evaluation algorithms. The feasibility of the approach is shown by experimental execution of pick-and-place tasks. The tasks are formulated as precedence graphs and may be carried out by a team of humans and a robot with a camera attached.

Currently, the implementation does not make use of the fact, that one camera image might provide information about the state of several conditions. This results in unnecessary robot motion, as the system does not remember the previously seen and performs a sensor operation for every evaluation request. However, it does not suffice to extract and store maximum information from one image. Due to unpredictable human actions, the state of conditions may change with time. In our future work, we will integrate a world model emulating the process of remembering and forgetting to deal with this problem of data aging.

References

1. M. Foster and C. Matheson, Following Assembly Plans in Cooperative, Task-Based Human-Robot Dialogue, Proc. of the 12th Workshop on the Semantics and Pragmatics of Dialogue, 2008
2. M. Fritzsche, N. Elkmann and E. Schulenburg, Tactile Sensing: A Key Technology for Safe Physical Human Robot Interaction, 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Lausanne, 2011
3. P. Ferreira, S. Doltsinis and N. Lohse, Symbiotic Assembly Systems – A New Paradigm, Procedia CIRP, vol. 17, pp. 26–31, 2014
4. T. Fong et al., The Human-Robot Interaction Operating System, Proc. of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, 2006
5. T. Gecks and D. Henrich, Sensor-Based Online Planning of Time-Optimized Paths in Dynamic Environments, GWR09 German Workshop on Robotics, Germany, 2009
6. T. Hamabe, H. Goto and J. Miura, A Programming by Demonstration System for Human-Robot Collaborative Assembly Tasks, IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, 2015
7. G. Hoffman and C. Breazeal, Collaboration in Human-Robot Teams, Proc. of the AIAA 1st Intelligent Systems Technical Conference, Chicago, 2004
8. L.S. Homem de Mello and A.C. Sanderson, Representations of Mechanical Assembly Sequences, IEEE Transactions on Robotics and Automation, vol. 7, no.2, 1991
9. N. Jennings, On Agent-Based Software Engineering, Artififfial Intelligence, Volume 117, Issue 2, 2000
10. L. Johannsmeier and S. Haddadin, A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes, IEEE Robotics and Automation Letters 2.1: 41-48, 2017
11. C. M. Jonker, M. B. van Riemsdijk and B. Vermeulen, Shared Mental Models: A Conceptual Analysis, COIN 2010 International Workshop, Lecture Notes in Artificial Intelligence Volume 6541, 2010
12. H. Kimura, T. Horiuchi and K. Ikeuchi, Task-Model Based Human Robot Cooperation Using Vision, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyongju, 1999
13. S. Lalle et al., Towards a Platform-Independent Cooperative Human Robot Interaction System: III An Architecture for Learning and Executing Actions and Shared Plans, IEEE Transactions on Autonomous Mental Development, vol. 4, no. 3, 2012
14. M. E. Latoschik and H. Tramberend: A Scala-Based Actor-Entity Architecture for Intelligent Interactive Simulations, 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012.
15. J. Laursen, U.Schultz and L. Ellekilde, Automatic Error Recovery in Robot Assembly Operations using Reverse Execution, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015
16. C. Lenz et al., Joint-Action for Humans and Industrial Robots for Assembly Tasks, 17th IEEE International Symposium on Robot and Human Interactive Communication, Munich, 2008
17. S. J. Levine and B. C. Williams, Concurrent Plan Recognition and Execution for Human-Robot Teams, 24th International Conference on Automated Planning and Scheduling (ICAPS), 2014
18. B. Lim and K. Klein, Team Mental Models and Team Performance: A Field Study of the Effects of Team Mental Model Similarity and Accuracy, Journal of Organizational Behavior, 27(4):403, 2006

19. E. Mathieu et al., The Influence of Shared Mental Models on Team Process and Performance, The Journal of Applied Psychology, 2000
20. G. Milliez et al., Using Human Knowledge Awareness to adapt Collaborative Plan Generation, Explanation and Monitoring, 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Christchurch, 2016
21. S. Nikolaidis and J. Shah, Human-Robot Cross-Training: Computational Formulation, Modeling and Evaluation of a Human Team Training Strategy, 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Tokyo, 2013
22. A. Schneider, J. Sturm et al., Object Identification with Tactile Sensors using Bag-of-Features, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, 2009
23. D. Schroeter et al., Methodology to Identify Applications for Collaborative Robots in Powertrain Assembly, Procedia CIRP, vol. 55, pp. 12–17, 2016
24. J. Shah et al., Improved Human-Robot Team Performance using Chaski, a Human-Inspired Plan Execution System, Proc. of the 6th International Conference on Human-Robot Interaction (HRI), Lausanne, 2011
25. U. Thomas, B. Finkemeyer, T. Kroger and F. M. Wahl, Error-Tolerant Execution of Complex Robot Tasks based on Skill Primitives, IEEE International Conference on Robotics and Automation (ICRA), Taipei, 2003
26. T. Werner and D. Henrich, Efficient and Precise Multi-Camera Reconstruction, 8th ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), Venice, 2014
27. T. Werner et al., ENACT: An Efficient and Extensible Entity-Actor Framework for Modular Robotics Software Components, 47th International Symposium on Robotics (ISR), 2016
28. H. Yussof et al., Low Force Control Scheme for Object Hardness Distinction in Robot Manipulation based on Tactile Sensing, IEEE International Conference on Robotics and Automation (ICRA), Pasadena, 2008

The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-662-54441-9_2.